TECHNICAL WHITE PAPER: November 2025

VMware Tanzu Greenplum on vSphere Kubernetes Service on VMware Cloud Foundation

Reference Architecture

Table of contents

Executive Summary

As enterprises increasingly adopt cloud-native platforms to support large-scale analytics, the need for a scalable, secure, and operationally efficient deployment model for data platforms has never been greater. Tanzu Greenplum on VMware vSphere Kubernetes Service (VKS) combines the proven performance of Greenplum's massively parallel processing (MPP) analytics engine with the agility and automation of Kubernetes. By leveraging the Greenplum Operator on Kubernetes, organizations can deploy, manage, and scale Greenplum clusters with significantly reduced setup time, optimized resource utilization, and built-in resiliency. This document outlines why Greenplum on VKS is an ideal choice for modern analytics workloads, explains its architecture and operational model, and demonstrates how it enables faster onboarding, agile development, and enterprise-grade stability.

Introduction

In today's data-driven world, scalable and flexible data infrastructure is foundational to analytics, AI, and business intelligence initiatives. Tanzu Greenplum has long been recognized as a powerful analytical MPP database capable of handling large-scale data processing across diverse workloads. Traditionally, deploying and operating Greenplum required provisioning multiple virtual machines, pre-configuring dependencies, and managing cluster lifecycles manually—often resulting in long deployment cycles and higher operational overhead.

Tanzu Greenplum on VMware vSphere Kubernetes Service modernizes this model by bringing Greenplum into a Kubernetes-native deployment paradigm. Using the Tanzu Greenplum Operator for Kubernetes, organizations can provision Tanzu Greenplum clusters directly on existing Kubernetes environments, dramatically reducing turnaround time and simplifying operations. This approach allows teams to focus on analytics and data value rather than infrastructure management, while still benefiting from Greenplum's performance, scalability, and advanced analytical capabilities.

Intended Audience

This document is intended for:

- **Platform and Infrastructure Engineers** responsible for Kubernetes, VMware environments, and resource management.
- **Data Platform and Database Administrators** who manage Greenplum clusters and require efficient lifecycle operations.
- **Data Engineers and Analytics Architects** designing scalable analytics platforms on Kubernetes.
- **DevOps and CI/CD Teams** looking to integrate analytics workloads into automated pipelines.
- **Enterprise IT and Cloud Architects** evaluating Kubernetes-based deployment models for analytics platforms.

A working knowledge of Kubernetes, VMware infrastructure, and analytical databases is helpful but not mandatory.

Document Purpose

The purpose of this document is to provide a comprehensive overview of Tanzu Greenplum on vSphere Kubernetes Service, covering both conceptual and practical aspects of deployment and operation. Specifically, it aims to:

- Explain why Greenplum on VKS is a compelling choice for modern, cloud-native analytics.
- Highlight benefits such as simplified operations, faster deployment, optimal resource consumption, security, stability, and scalability.
- Describe the architecture and working model of Greenplum deployed via the Kubernetes Operator.
- Walk through setup and deployment on a VKS environment, including cluster creation, access, and monitoring.
- Explain failure detection, automated recovery, and cluster resiliency.
- Showcase real-world use cases, including customer onboarding, agile development and testing, CI/CD automation, and strong isolation with minimal resources.

By the end of this document, readers should have a clear understanding of how Greenplum on VKS enables faster, more efficient, and more resilient analytics platforms in Kubernetes-based enterprise environments.

Benefits of using Greenplum on VKS

While Greenplum on Kubernetes delivers significant advantages across any Kubernetes platform, deploying on vSphere Kubernetes Service (VKS) provides unique enterprise-grade capabilities that address the specific needs of large-scale, production-ready analytical database deployments. The following sections highlight the key benefits that Greenplum makes from VKS for its workloads.

Secure Airgap Deployment Without Compromising Agility: For organizations operating in air-gapped or highly regulated environments, Greenplum requires the exposure of the coordinator host and Greenplum command center to enable users to access and use Greenplum clusters and GPCC. With the availability of NSX in VKS, Greenplum Database (GPDB) and Greenplum Command Center (GPCC) clusters can be exposed easily and in a secured way to the end users in a regulated way. This capacity holds considerable significance for governmental bodies, financial institutions, healthcare providers, and defense contractors that are mandated to uphold stringent network isolation while concurrently leveraging modern analytical database technologies.

Dedicated Workload Zones for Optimal Greenplum Performance: Since Greenplum is a massively parallel processing database, customers target to deploy it on a large scale (like 50 - 100 nodes) with high memory and CPU requirements to run their workloads and leverage Greenplum. VMware Cloud Foundation 9.0 introduces enhanced flexibility through VKS workload zones, enabling Admins to define and manage specialized infrastructure zones that align precisely with Greenplum's performance and resource requirements. This capability allows organizations to create dedicated zones optimized for specific analytical workloads. By dedicating resources to Greenplum workloads, organizations can ensure consistent performance and eliminate resource contention from other applications running in the same Kubernetes cluster.

Unified Management for Enterprise-Scale Greenplum Deployments: As organizations scale their Greenplum deployments across multiple environments—development, testing, staging, and production—the complexity of managing numerous clusters grows exponentially. vSphere Kubernetes Service addresses this challenge through VKS Cluster Management integration, which provides Cloud Admins with unified oversight and control over all Greenplum clusters regardless of their location or environment. This fleet-wide multi-cluster management capability enables teams to view, configure, and manage clusters from a single interface, ensuring consistent policies, security controls, and operational procedures across the entire Greenplum infrastructure. Since VKS operates containers at the same level as VMS, it would be easier for the administrators to control the workloads and manage them.

Simplified Operations and High-Performance Infrastructure: Greenplum requires large and high performant storage capabilities for their workloads to run efficiently. VKS simplifies this task through the out-of-the-box high-performance shared storage through vSAN, which provides persistent storage policies tailored to stateful Kubernetes workloads like Greenplum databases. This integration ensures that Greenplum clusters benefit from the same enterprise-grade resiliency features as virtual machines, including vSphere HA, and seamless recovery. The supervisory cluster architecture enables organizations to leverage existing infrastructure investments while gaining the agility and scalability benefits of Kubernetes, making Greenplum deployment as straightforward as provisioning a virtual machine, but with the added advantages of container orchestration, automated scaling, and modern DevOps practices.

More Speed Less Turn Around Time: The deployment time for Greenplum Clusters has been significantly reduced. This automated process ensures error-free provisioning of clusters, which is particularly beneficial for short-lived projects, Proof of Concepts (POCs), and experimental endeavors. Teams are now empowered with the capability to launch Greenplum cluster environments rapidly, whether for prototypes or larger workloads.

Optimal Resource Consumption: The Greenplum on Kubernetes offers significant advantages for data management and analytics. This integration empowers users to deploy new Greenplum clusters with remarkable efficiency, requiring only minimal resource allocation. This is particularly beneficial for organizations looking to scale their data warehousing capabilities on demand without incurring substantial upfront infrastructure costs.

Beyond new deployments, a key strength of this synergy lies in its ability to leverage existing Kubernetes environments for GPDB provisioning. This optimizes resource utilization by allowing GPDB instances to share the underlying infrastructure with other applications and services already running within Kubernetes. The result is a substantial reduction in hardware and operational requirements, leading to cost savings and improved environmental sustainability.

Security and Segregation: The advent of Kubernetes introduces enhanced Role-Based Access Control (RBAC), enabling diverse deployments within a single cluster through distinct namespaces. The implementation of robust access controls mitigates the "blast radius" of security incidents, thereby streamlining compliance by establishing clearly defined and auditable environments. Extensive Kubernetes features such as Security Context, Node Affinity, and Pod Anti-affinity can facilitate the implementation of best practices during Greenplum deployment.

Stability: In Kubernetes-based Greenplum environments, stability extends far beyond simple uptime. The platform ensures complete automated recovery, as failed pods are seamlessly restarted and reattached to their existing persistent volumes—eliminating manual intervention and preserving data integrity. Resources can be pre-allocated, so Greenplum always operates with the compute and memory necessary for peak analytics performance, guaranteeing there's no degradation even as workloads scale. The use of robust, predefined upgrade and patching procedures further enhances platform reliability, ensuring that improvements or security fixes are applied safely and predictably.

Scalability: Depending on the cluster's requirements, allocated resources can be scaled to ensure uptime and automated expansion, as deploying pods in stateful sets performs rolling updates. The segregation of compute and storage is highly beneficial in ensuring that pods are scaled only as required, depending on performance or data volume.

Lower TCO: With VKS organizations have the ability to reduce silos, leverage existing tools and skill sets without having to retrain staff and/or change existing processes. Utilizing unified lifecycle management across infrastructure components to stay up-to-date with the most recent patches and minimizing security risks.

Operational Simplicity: VKS is engineered for unparalleled operational simplicity, leveraging the familiarity of existing vSphere tools, skills, and workflows. This design philosophy significantly reduces the learning curve for IT teams and streamlines management processes. With VKS, organizations benefit from automated cluster provisioning, which accelerates deployment times and minimizes manual configuration errors. Furthermore, its robust capabilities extend to automated upgrades and comprehensive lifecycle management. This integrated approach ensures consistent operations, reduces overhead, and frees up valuable resources to focus on innovation rather than infrastructure maintenance.

Use Cases

Customer Onboarding

A significant challenge encountered during Greenplum customer onboarding is the initial learning curve, along with the requisite configurations and operations, that must be completed before the full features and capabilities of Greenplum can be leveraged. This extended setup period can prolong the database evaluation process, thereby increasing the time required for customer onboarding, particularly for those migrating from alternative database systems.

The integration of Greenplum on Kubernetes significantly reduces this timeframe, providing customers with access to Greenplum through a minimal set of steps and limited prerequisites, thereby facilitating a smooth and hassle-free onboarding process. For individuals demonstrating Greenplum features, deployment is achievable even on a local system utilizing a minikube cluster. This capability is a crucial step in customer onboarding and for field personnel, as it enables minimal and streamlined deployments.

Agile Development and Testing Environments

The advent of Greenplum on Kubernetes facilitates data teams in expeditiously establishing isolated Greenplum clusters, meticulously configured for developmental, testing, and proof-of-concept initiatives. This functionality guarantees that each team can operate within a pristine, standardized environment on demand, thereby accelerating innovation cycles and mitigating the potential for disruptions to production systems.

CI/CD Workload and Automation

For users seeking to deploy a Greenplum Database cluster for testing and analytical endeavors, and to leverage Greenplum within their operational pipelines, the implementation of Greenplum on Kubernetes offers substantial advantages in deployment. This approach also mitigates the risks associated with manual lifecycle operations, thereby enhancing system stability.

Strong Isolation on Clusters on Minimal Resource

With the enforcement of best Kubernetes practices and the GP on Kubernetes introduction, we can leverage a single Kubernetes cluster to run different kinds of Greenplum instances isolated from each other. We can organize Greenplum instances into various namespaces and can enforce things like resource quota to ensure optimal resource utilization and segregation.

vSphere Kubernetes Service

vSphere Kubernetes Service (VKS) is the Kubernetes runtime built directly into VMware Cloud Foundation (VCF). With CNCF certified Kubernetes, VKS enables platform engineers to deploy and manage Kubernetes clusters while leveraging a comprehensive set of cloud services in VCF. Cloud admins benefit from the support for N-2 Kubernetes versions, enterprise grade security, and simplified lifecycle management for modern apps adoption.

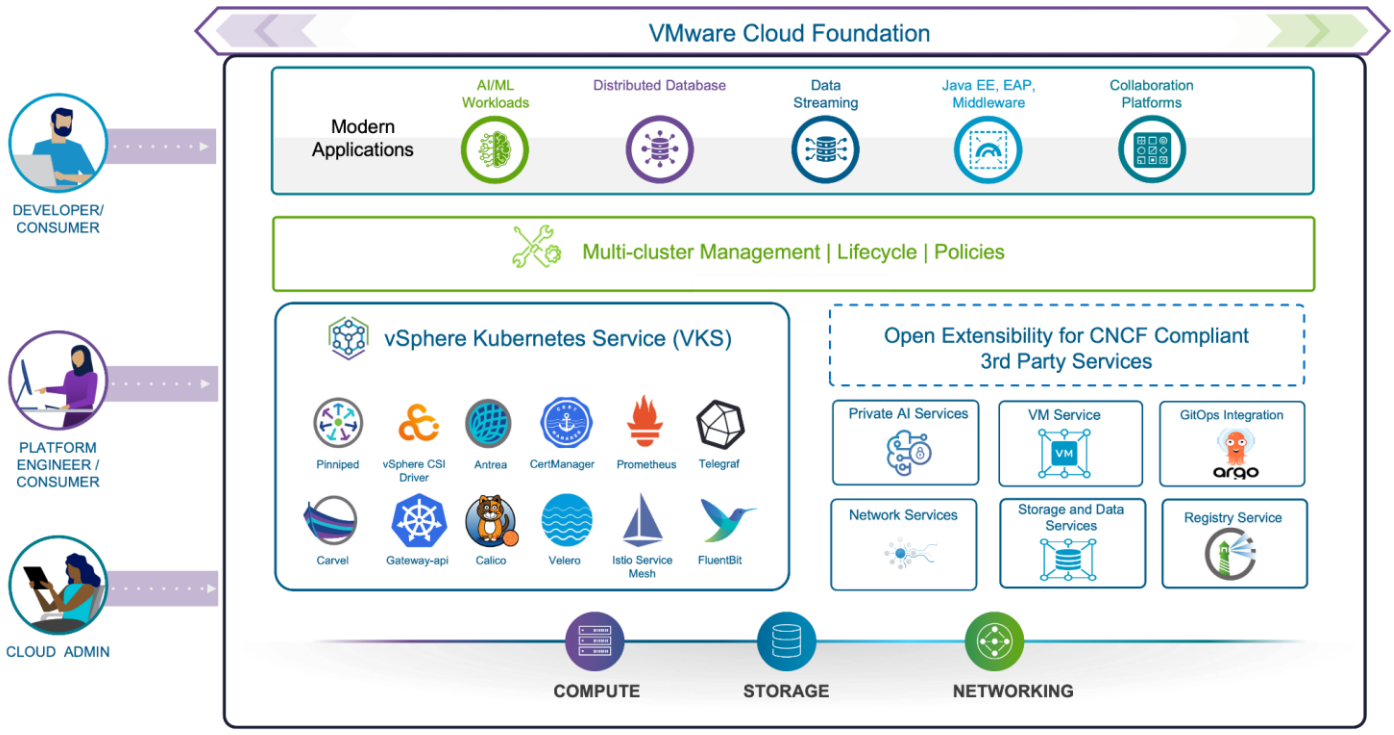

Figure 1: VKS on VCF Ecosystem

Key Architectural Greenplum Components

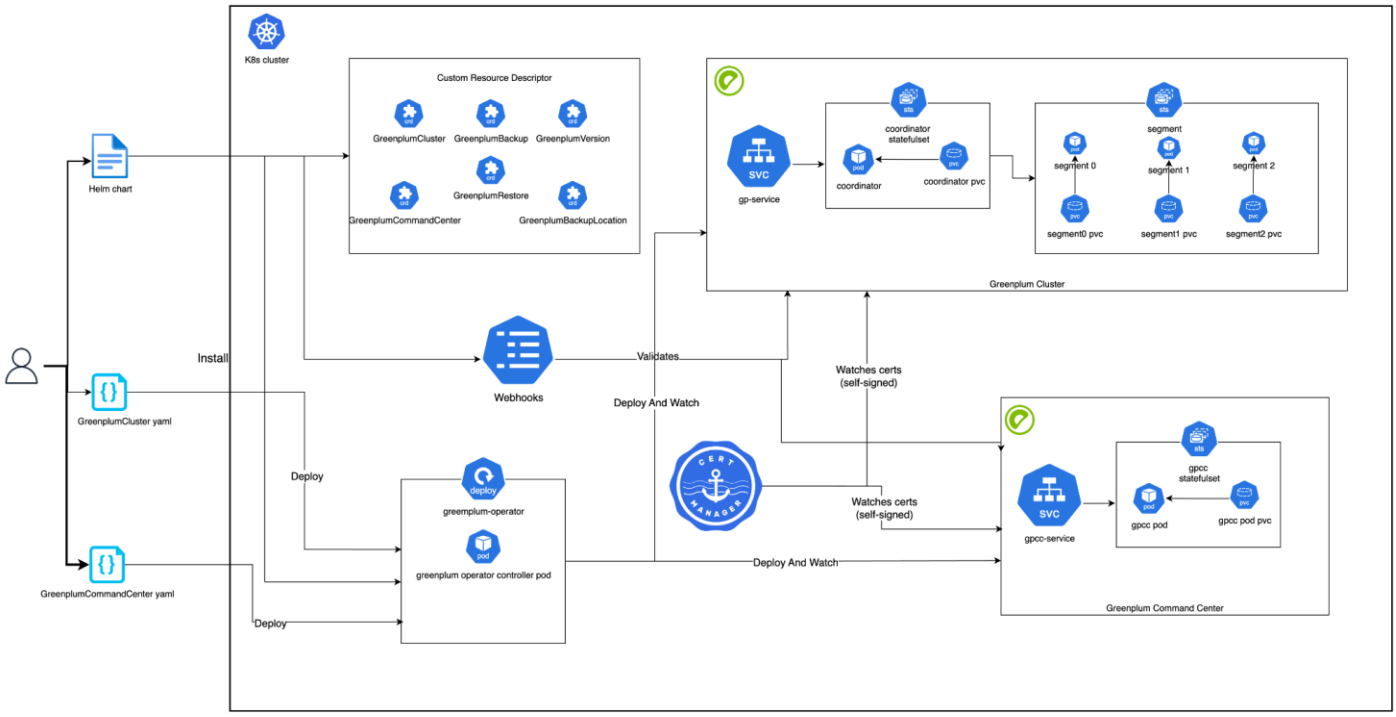

Figure 2: Greenplum Architecture Overview

- **Greenplum Operator**: Greenplum operator is the key controller which is responsible for managing the Greenplum cluster on Kubernetes, including deployment, lifecycle management, etc. It interacts with Kubernetes resources to configure and manage the cluster.
- **Greenplum Cluster**: This is a fully provisioned and manageable GPDB cluster over Kubernetes represented by a Greenplum Cluster Custom Resource (CR). The CR is a combination of coordinator and segment deployments along with an exposing service to interact with the cluster.
- **Greenplum Version**: This is a Custom Resource which defines the GPDB version and its image details. This is used in the Greenplum Cluster CR to define the GPDB Cluster Version.
- **Greenplum Command Center**: This is a Custom Resource which will provision a Greenplum Command Center Instance associated with a Greenplum Cluster
- **Cert Manager**: It is required and used for watching self-signed certs used by the operator and webhooks
- **Webhooks**: These are deployed to validate the gp-operator provisioned resources like greenplum clusters, greenplum commandcenters, etc.

Quick Overview of the install:

- User first installs cert-manager if not present. This watches over self-signed certificates and webhooks used by the operator.
- Upon the installation of cert-manager, the user proceeds to deploy the GP operator helm chart. Helm chart consists of various CRDs and deployment of the operator.
- Upon the deployment of the operator, the user subsequently creates the Greenplum version resource, which encapsulates details regarding the Greenplum and Command Center instances slated for provisioning.
- Once the Greenplum version is created, users then create Greenplum cluster and Greenplum command center resources based on their requirements which are then validated by webhooks.
- The operator creates the required resources and spins up the clusters. Operator also watches and reconciles the failed instances and brings them to appropriate state.
- Upon cluster activation, users can access it through the pods or services deployed as a part of the cluster.

Solution Requirements

Deploying Greenplum on Kubernetes in a VMware vSphere Kubernetes Service (VKS) environment enables scalable, cloud-native analytics with container orchestration. In order to deploy and manage Greenplum databases in a VKS-managed Kubernetes cluster, including operator-based lifecycle management, persistent storage, and integration with Kubernetes-native services we need to ensure that the VKS cluster meets the following prerequisites:

- A storage class to be referred by the persistent volumes of Greenplum instance, cert-manager must be installed and configured to handle TLS certificates for secure communication between Greenplum components.
- Optionally, configure an S3-compatible object storage service (such as MinIO or AWS S3) that is accessible from the cluster to enable backup and restore operations for your Greenplum databases.
- Additionally, you will need access to an image registry; either the official registry or a self-hosted registry that contains the required Greenplum container images, including the Greenplum operator, instance images, and the "greenplum on kubernetes" Helm chart.
- Please refer to official documentation for downloading the Tanzu Greenplum Kubernetes package: https://techdocs.broadcom.com/us/en/vmware-tanzu/data-solutions/tanzu-greenplum-k8s/1-0/tgp-on-k8s/04-installation.html

Create your own Greenplum Cluster

The deployment of Greenplum Clusters on Kubernetes has been significantly simplified with the introduction of the operator. Users can provision GPDB clusters from the following minimal steps:

Install the helm chart:

```
# Install cert manager (if not installed already). Sample
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.18.2/cert-manager.yaml

# Download the helm chart and install the operator
# Login to the helm registry and pull the chart. Example
```

```
helm registry login <URL> -u ${USER} -p ${PASSWORD}
helm pull oci://<CHART_PATH>/gp-operator --version 1.0.2

# Untar the chart and change the directory
tar -xvzf gp-operator-1.0.2.tgz
cd gp-operator

# Edit the secret in values.yaml file
imagePullSecrets:
   - name: <YOUR_SECRET_NAME>
 # Make sure you are in the gp-operator directory
helm install gp-operator . -n ${NAMESPACE}
```

Alternative to the previous step, you can download the unified deployer tar and extract it to host images in a self-hosted image registry and deploy the helm chart:

```
tar -xzvf greenplum-for-kubernetes-1.0.2.tar.gz

# Use the deployer.sh script to automate the deployment process. The script can push the necessary container
images to a private registry and deploy the official Helm chart.

# Sample to push images and deploy helm chart using the pushed image is as follows

./deployer.sh --action all --registry-url your-registry.com/gp-images --namespace greenplum --requires-auth y
```

Once the helm chart is deployed, create Greenplum Version using greenplumversion custom resource. This will provide the details needed to provision a greenplum cluster. Sample file for the greenplumversion is shown below:

```
# vi gpversion7.4.1.yaml
# Sample for Greenplum Version 7.4.1
apiVersion: greenplum.data.tanzu.vmware.com/v1
kind: GreenplumVersion
metadata:
  labels:
    app.kubernetes.io/name: gp-operator
    app.kubernetes.io/managed-by: kustomize
  name: greenplumversion-7.4.1
spec:
  dbVersion: 7.4.1
  image: <GPDB_IMAGE_PATH>/gp-instance:7.4.1
  operatorVersion: 1.0.2
  extensions:
    - name: postgis
      version: 3.3.2
    - name: greenplum_backup_restore
      version: 1.31.0
  gpcc:
    version: 7.4.0
    image: <GPCC_IMAGE_PATH>/gp-command-center:7.4.0

# Create greenplum version
kubectl apply -f gpversion7.4.1.yaml
```

Once the helm chart is deployed, create Greenplum Version using greenplumversion custom resource. This will provide the details needed to provision a greenplum cluster. Sample file for the greenplumversion is shown below:

```
# vi gpversion7.4.1.yaml
# Sample for Greenplum Version 7.4.1
apiVersion: greenplum.data.tanzu.vmware.com/v1
kind: GreenplumVersion
```

```
metadata:
  labels:
    app.kubernetes.io/name: gp-operator
    app.kubernetes.io/managed-by: kustomize
  name: greenplumversion-7.4.1
spec:
  dbVersion: 7.4.1
  image: <GPDB_IMAGE_PATH>/gp-instance:7.4.1
  operatorVersion: 1.0.2
  extensions:
    - name: postgis
      version: 3.3.2
    - name: greenplum_backup_restore
      version: 1.31.0
  gpcc:
    version: 7.4.0
    image: <GPCC_IMAGE_PATH>/gp-command-center:7.4.0

# Create greenplum version
kubectl apply -f gpversion7.4.1.yaml
```

Once the greenplumversions are created, we can create greenplumclusters. Sample file to create greenplumcluster is shown below:

```
# vi gp-minimal.yaml

apiVersion: greenplum.data.tanzu.vmware.com/v1
kind: GreenplumCluster
metadata:
  labels:
    app.kubernetes.io/name: greenplum-cluster-operator
    app.kubernetes.io/component: greenplum-operator
    app.kubernetes.io/part-of: gp-operator
  name: gp-minimal
spec:
  version: greenplumversion-7.4.1
  imagePullSecrets:
    - mds-gar-secret # optional
  coordinator:
    storageClassName: standard
    service:
      type: NodePort
    storage: 1Gi
  global:
    gucSettings:
      - key: wal_level
        value: logical

  segments:
    count: 1
    storageClassName: standard # Update this to use your storageClass
    storage: 10Gi # Storage per segment

# Create greenplum cluster
kubectl apply -f gp-minimal.yaml

# Check greenplum cluster status

kubectl get greenplumclusters
# OR
kubectl get gp

NAME              STATUS    AGE
gp-minimal        Running   3d20h
```

Once the Greenplum Cluster is deployed you can also deploy Greenplum Command Center as a monitoring tool for this cluster. Sample for GPCC deployment is shown below:

```
# vi gpcc-minimal.yaml
apiVersion: greenplum.data.tanzu.vmware.com/v1
kind: GreenplumCommandCenter
metadata:
  labels:
    app.kubernetes.io/name: gp-operator
    app.kubernetes.io/managed-by: kustomize
  name: gpcc
spec:
  storageClassName: standard
  greenplumClusterName: gp-minimal
  storage: "2Gi"

# Create your Greenplum Command Center Instance
kubectl apply -f gpcc-minimal.yaml -n <NAMESPACE>

# Check GPCC Status
kubectl get gpcc
```

Accessing GPDB and GPCC Clusters

Accessing Greenplum Cluster

We can access the greenplum cluster in one of the following ways:

- If greenplum service is provisioned with serviceType as LoadBalancer, then we can directly access it using the LoadBalancerIP. This is the recommended approach.
- We can port-forward greenplum-coordinator pod and access it locally using psql or any other client.
- We can port-forward greenplum svc and access it locally using psql or any other client. This approach can be used for demo and local testing.We can exec into greenplum-coordinator pod and access it locally using psql or any other client.
- Once the cluster is accessible you can use psql to perform DB Operations or any other tool. You can get the cluster credentials by following the step below:

```
# make sure tha the cluster is in RUNNING state
kubectl get gp -n ${NAMESPACE}
# kubectl get gp -n gpdb
# NAME    STATUS    AGE
# gp-minimal        Running   3d20h

# Fetch User Password
# Fetch User Credential Secret of gp
kubectl get secret -n ${NAMESPACE} GP_INSTANCE_NAME-creds -o yaml

# Example
kubectl get secret -n ${NAMESPACE} gp-minimal-creds -o yaml

# Sample Data
# you will get gpadmin as key and base64 encoded password as value
gpadmin:QWRtaW4xMjMK

# Take the value and perform base64 decode to get the user password
echo 'QWRtaW4xMjMK' | base64 -d


# Port forward greenplum svc
kubectl port-forward -n ${NAMESPACE} svc/<CLUSTER_NAME>-svc 8080:5432
```

```
# Port forward greenplum coordinator pod
kubectl port-forward -n ${NAMESPACE} pod/<CLUSTER_NAME>-coordinator-0 8080:5432

# Exec into greenplum coordinator pod
kubectl exec -it  <CLUSTER_NAME>-coordinator-0 -n ${NAMESPACE} --/bin/bash


# Use psql to connect to the instance
# Sample
psql -U gpadmin -d postgres -H HOSTNAME -p PORT -w PASSWORD
# When you Port forward greenplum svc
psql -U gpadmin -d postgres -H localhost -p 8080 -w PASSWORD
```

Accessing GPCC Instance

There are many ways to access the provisioned GPCC:

- If the service is provisioned using the LoadBalancer service type, we can directly hit the LoadBalancer IP to access the instance.
- Port forward the GPCC pod
- Port forward the GPCC service

```
# Sample Port forward GPCC pod
kubectl port-forward pod/gpcc-cc-app-0 -n ${NAMESPACE} 8080:8080

# Sample Port forward GPCC service
kubectl port-forward svc/gpcc-cc-svc -n ${NAMESPACE} 8080:8080
```

Fetching Username and Password for GPCC

Once we can access the GPCC instance, we would be required to perform login on that instance. While deploying GPCC, we create a user ‘gpmon’ for performing GPCC related operations on DB and to use GPCC. To fetch the username and password perform the following:

```
# Fetch User Credential Secret of gpcc
kubectl get secret -n ${NAMESPACE} GPCC_INSTANCE_NAME-cc-creds -o yaml

# Example
kubectl get secret -n ${NAMESPACE} gpcc-cc-creds -o yaml

# Sample Data
# you will get gpmon as key and base64 encoded password as value
gpmon:QWRtaW4xMjMK

# Take the value and perform base64 decode to get the user password
echo 'QWRtaW4xMjMK' | base64 -d

# To change the user password, edit the secret and change the value of gpmon key
# note that value should be in base64  encoded format
# gpmon password will be altered in DB too
kubectl edit secret -n ${NAMESPACE} gpcc-cc-creds
# Sample
gpmon:QWRtaW4xMjM0NQo=
```

Greenplum Cluster Recoverability from failure

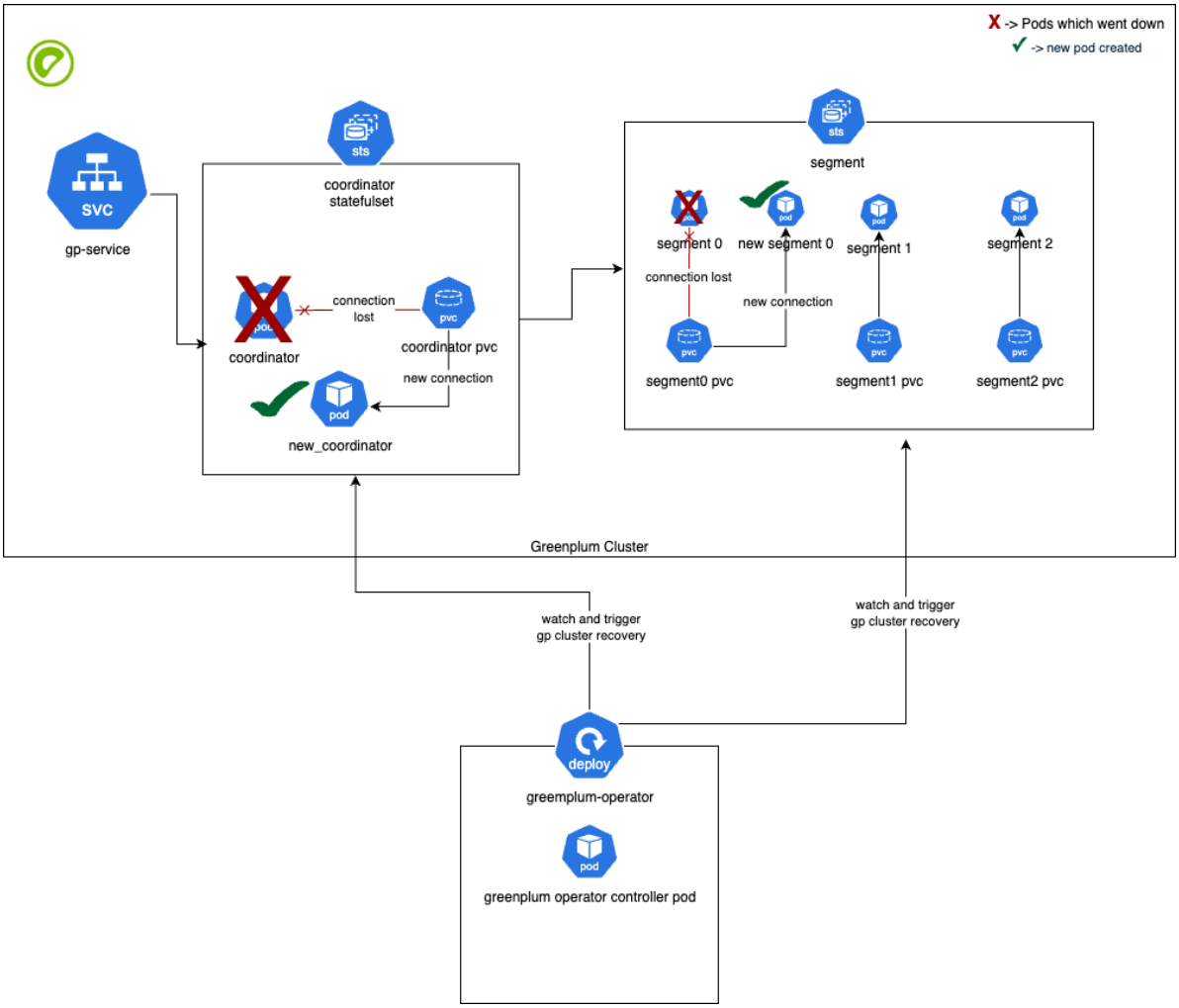

Figure 3: Greenplum Recovery Architecture

Automated Failure Detection and Recovery Workflow

If any component failure occurs, like coordinator or segment pod crashes due to software issues, or the underlying Kubernetes node becomes unavailable; the robust recovery mechanism ensures that your data remains safe and accessible.

- Kubernetes detects any unhealthy, unresponsive, or lost pod. In parallel, the Greenplum Operator, through its constant observation mechanism, identifies the change in the cluster's state.
- On detecting the failure, a new pod for the affected component is created.
- The original Persistent Volume (PV) and its associated Persistent Volume Claim (PVC), which holds all the Greenplum data for that component, is attached to the newly spun-up pod
- Once the pod is up and running and has its data volume securely attached, the Greenplum Operator triggers the necessary Greenplum-specific recovery tasks. This ensures that the cluster is restored to its original healthy state

Component	Version	Notes

Tanzu Greenplum on Kubernetes	1.0.2	
Tanzu Greenplum	7.4.1	
vSphere Kubernetes Service	3.4	
vSphere Cloud Foundation	9.0	
Kubernetes	1.33	

Table 1: Greenplum Validated Versions

Conclusion

This thorough validation of Tanzu Greenplum on VMware vSphere Kubernetes Service (VKS) solidifies the ability to easily deploy and operate a large data platform on VMware vSphere Kubernetes Service on VMware Cloud Foundation. Thus, delivering a highly scalable and cost-effective data platform solution, leveraging the inherent elasticity of Kubernetes on the VMware Cloud Foundation for optimized Total Cost of Ownership.

Item No: vmw-bc-wp-tech-temp-a4-word-2024 1/24